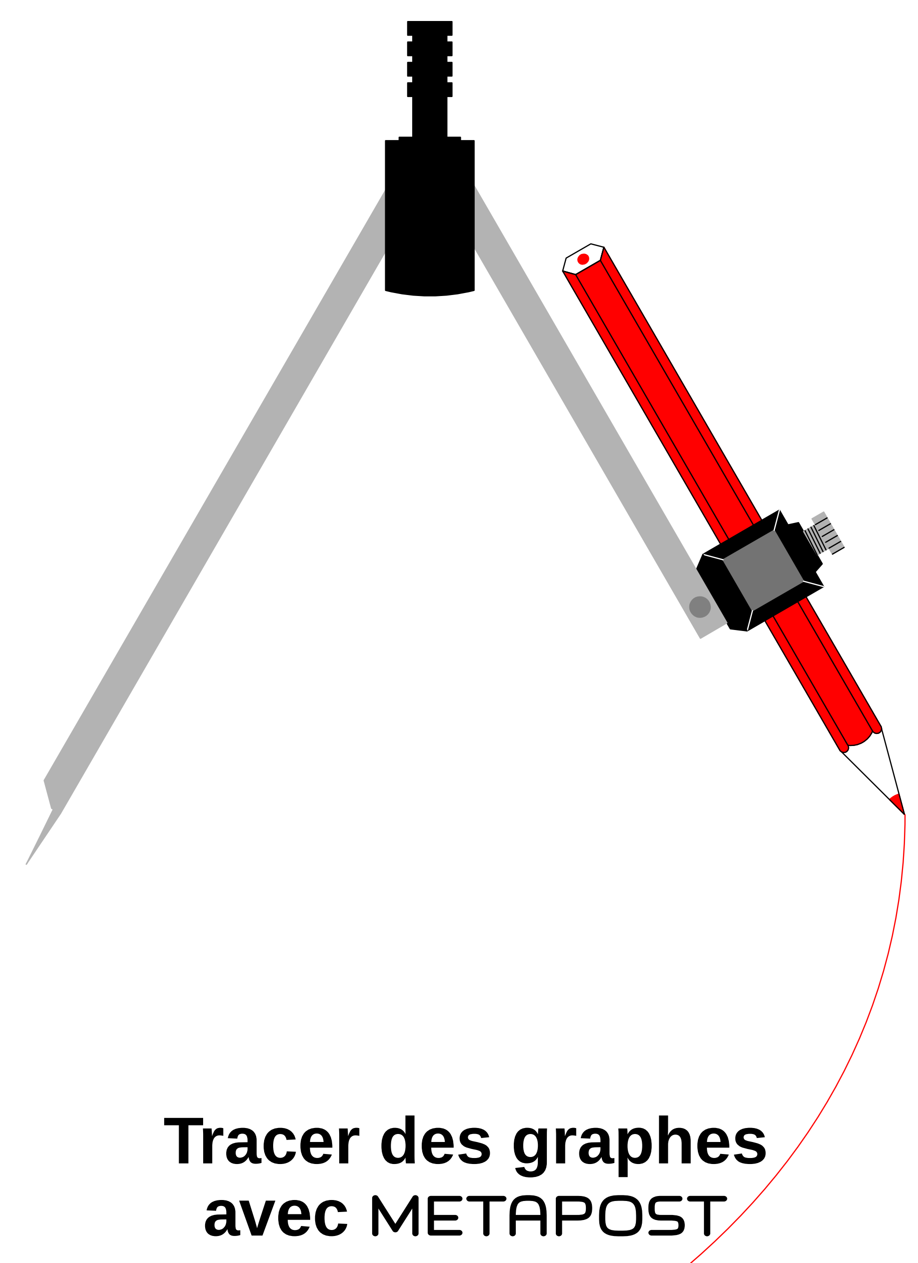


Tracer des graphes avec METAPOST

John D. HOBBY
Bell Labs, Lucent Technologies
Murray Hill, NJ 07974

Traduction
Pierre FOURNIER (pierre.fournier@unilim.fr)
Jean-Côme CHARPENTIER (Jean-Come.Charpentier@wanadoo.fr)

Table des matières

Résumé

Ce papier décrit un *package* de tracé de graphes qui a été implémenté comme une extension au langage graphique METAPOST. METAPOST possède des macros puissantes pour implémenter de telles extensions. Il existe également quelques nouvelles caractéristiques du langage qui supportent des macros de graphes. Les spécificités existantes pour produire et manipuler des figures permettent à l'utilisateur de faire des choses qui auraient été difficiles à réaliser avec un *package* de graphes autonome.

Abstract

This paper describes a graph-drawing package that has been implemented as an extension to the METAPOST graphics language. METAPOST has a powerful macro facility for implementing such extensions. There are also some new language features that support the graph macros. Existing features for generating and manipulating pictures allow the user to do things that would be difficult to achieve in a stand-alone graph package.

1 Introduction

METAPOST est un langage « par lot » orienté graphique, basé sur METAFONT de KNUTH, mais produisant des sorties POSTSCRIPT et possédant de nombreuses caractéristiques pour l'intégration de textes et de graphiques. L'auteur a essayé de rendre ce papier aussi indépendant que possible du manuel d'utilisation[5], mais quelques connaissances du langage METAPOST permettront d'apprécier toutes les fonctionnalités graphiques. Ce document se focalise sur les mécanismes de production de certains graphiques particuliers parce que la question de savoir quel est le meilleur type de graphique pour une situation donnée est couverte dans de nombreux ouvrages ; par exemple, CLEVELAND[2, 3, 4] et TUFTE[11]. La finalité est de fournir au moins la puissance de « grap » sous Unix [1], mais au moyen du langage METAPOST. Désormais le *package* est implémenté en utilisant la facilité des puissantes macros de METAPOST.

Les macros graph procurent les fonctionnalités suivantes :

1. échelles automatiques ;
2. génération automatique des graduations et de l'étiquetage des axes et/ou des quadrillages ;
3. systèmes de coordonnées multiples ;
4. échelles linéaires et logarithmiques ;
5. possibilité de manipuler des nombres en-dehors des gammes habituelles ;
6. symboles de traçage arbitraires ;
7. commandes de traçage, remplissage et étiquetage des graphes.

En plus de ces caractéristiques, l'utilisateur a également accès à toutes les caractéristiques décrites dans le manuel d'utilisation de METAPOST. Celles-ci incluent l'accès à presque toutes les caractéristiques du POSTSCRIPT : la possibilité d'utiliser et de manipuler du texte composé, la possibilité de résoudre des équations linéaires et l'utilisation des types de données pour les points, les courbes, les images et les transformations de coordonnées. La section 2 décrit les macros `graph` du point de vue de l'utilisateur et présente plusieurs exemples. Les sections 3 et 4 discutent des *packages* auxiliaires pour la manipulation et la composition des nombres. La section 5 présente quelques remarques en guise de conclusion. L'annexe A résume les macros de tracé de graphes et l'annexe B décrit quelques ajouts récents au langage METAPOST qui n'ont été présentés nulle part.

2 Utilisation des macros graph

Un fichier source METAPOST qui utilise les macros de graphe doit commencer par

```
input graph;
```

Cette commande permet de lire le fichier de macros `graph.mp` et définit les commandes de tracé de graphes qui sont expliquées dans la suite. Le reste du fichier doit être composé d'une ou plusieurs instances de

```
beginfig(<numéro de figure>) ;
<commandes de graphiques>
endfig;
```

suivies par `end`.

Les <commandes de graphiques> qui suivent, suffisent à produire le graphe de la figure 1 à partir du fichier de données `agepop91.d` :

```
draw begingraph(3in,2in) ;
gdraw"agepop91.d" ;
endgraph ;
```

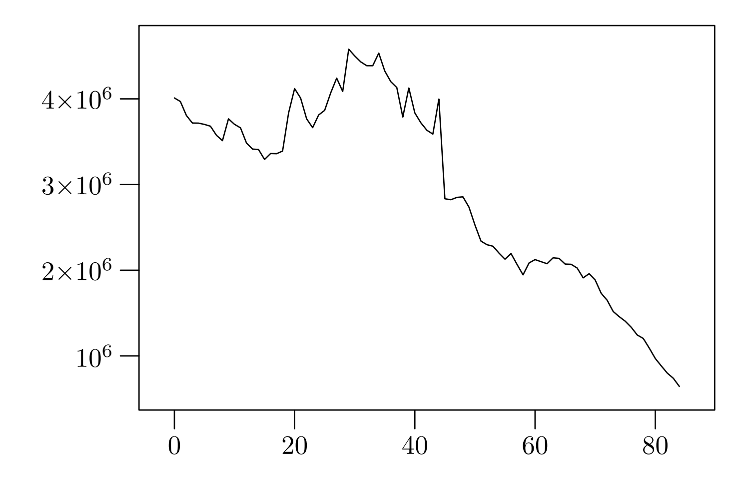


FIG. 1 – Graphique de la distribution d'âge en 1991, aux États Unis

2.1 Commandes élémentaires de tracé de graphes

Tous les graphes doivent commencer avec

`begingraph`(<largeur>,<hauteur>) ;

et se terminer avec `endgraph`. Cela est, d'un point de vue syntaxique, une <expression figure>, aussi cette instruction doit-elle être précédée par `draw` et suivie par un point-virgule, comme dans l'exemple[1]. La <largeur> et la <hauteur> donnent les dimensions du graphe lui-même sans les étiquettes d'axe. La commande

`gdraw`<expression><liste d'options>

dessine un graphe au trait. Si l'<expression> est de type chaîne de caractères, elle indique le nom d'une fichier de données ; sinon c'est un chemin qui donne la fonction à dessiner. La <liste d'options> contient une ou plusieurs options de dessin. Elle peut également être vide. Ces options sont

`withpen`<expression plume> — `withcolor`<expression couleur> —
`dashed`<expression figure>

qui indiquent l'épaisseur de ligne, la couleur ou le motif de trait comme l'explique le manuel d'utilisation [5].

En plus des options standard de dessin, la <liste d'options> dans la déclaration `gdraw` peut contenir

`plot`<expression dessin>

L'<expression dessin> indique le symbole de tracé qui sera employé à chaque nœud du chemin.

[1] Voir le manuel d'utilisation [5] pour des explications des commandes `draw` et des éléments de syntaxe tels que les <expressions images>.

L'option `plot` supprime le dessin de la ligne de telle manière que[2]

```
gdraw "agepop91.d" plot btex $\bullet$etex
```

produit seulement les points noirs (*bullets*) comme l'illustre la figure 2. Si l'on ajoute à l'option de dessin, l'option `withpen`, on obtient la superposition de la ligne sur les symboles de traçage.

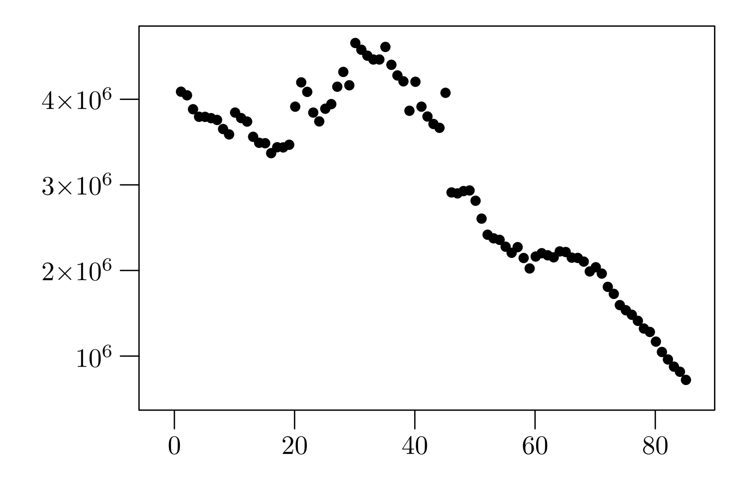


FIG. 2 – La distribution d'âge en 1991 dessinée avec des points noirs

Les commandes `glabel` et `gdotlabel` ajoutent des étiquettes sur le graphe. La syntaxe pour `glabel` est

`glabel`.<suffixe d'étiquette>(<expression chaîne ou dessin>,<position>)<liste d'options>

où la <position> identifie la localisation d'une étiquette et le <suffixe d'étiquette> précise le décalage de l'étiquette relative à cette position. La commande `gdotlabel` est identique à la précédente, excepté le fait que la position est marquée par un point. Un <suffixe label> s'écrit comme en `plain` METAPOST. Il peut être <vide> pour centrer l'étiquette à la position ; lft, rt, top, bot décalent l'étiquette horizontalement ou verticalement et `ulft`, `urt`, `llft`, `lrt` indiquent les décalages diagonaux. La <position> peut être un couple de coordonnées du graphe, un numéro de nœud du dernier chemin tracé avec gdraw ou la position spéciale `OUT`. Ainsi

```
gdotlabel.top(btex $(50,0)$ etex, 50,0)
```

marquera un point aux coordonnées (`50,0`) et placera le texte composé « (50,0) » au-dessus de ce dernier. De façon similaire,

```
glabel.ulft("Knot 3",3)
```

composera la chaîne `"Knot 3"` et la placera au-dessus et à gauche du nœud 3 du dernier chemin obtenu par la commande `gdraw`.

[2] Les utilisateurs de Troff doivent remplacer `btex $\ bullet$ etex` par `btex \(bu etex`.

La <position> `OUT` place l'étiquette relativement à l'ensemble du graphe. Par exemple, remplacer « `gdraw "agepop91.d"` » par

```
glabel.lft(btex \vbox{\hbox{Population}
\hbox{en millions}} etex, OUT) ;
glabel.bot(btex Âge en années etex, OUT) ;
gdraw "apepopm.d" ;
```

dans le fichier source de la figure 1 produit la figure 3. Ceci améliore le graphe en ajoutant des noms d'axes et en utilisant un nouveau fichier de données `agepopm.d` où les chiffres de la population ont été divisés par 1 million pour éviter les valeurs trop grandes. On verra plus loin que de simples transformations telles que celles-ci peuvent être obtenues sans avoir à produire de nouveaux fichiers de données.



FIG. 3 – Une version améliorée du graphe de la distribution d'âge en 1991

Les possibilités de TEX permettent de gérer des étiquettes sur plusieurs lignes avec la commande `\hbox` à l'intérieur de commandes `\vbox` utilisés dans l'exemple précédent. Cependant les utilisateurs de LATEX trouveront sans doute plus naturel de recourir à l'environnement `tabular` [9]. Les utilisateurs de troff peuvent utiliser

```
btex .nf
Population en millions
etex
```

2.2 Systèmes de coordonnées

Les macros graphiques du *package* `graph` décalent et remettent automatiquement à l'échelle les coordonnées à partir des fichiers de données et les chemins tracés avec `gdraw` ainsi que les positions `glabel` pour tenir dans le graphique.

Que l'amplitude de l'ordonnée y soit de 0.64 à 4.6 ou de 460 000 à 4 600 000, les valeurs se répartiront de façon à remplir environ 88 % de la hauteur spécifiée dans l'instruction `begingraph`. Naturellement les épaisseurs de lignes, les étiquettes et les symboles de tracé ne sont pas remis à l'échelle.

La commande `setrange` contrôle le processus de décalage et de mise à l'échelle en spécifiant le minimum et le maximum des coordonnées :

`setrange`(<coordonnées>,<coordonnées>)

où
<coordonnées> *to* <expression couple>
| <expression numérique ou chaîne>,<expression numérique ou chaîne>

Le premier couple <coordonnées> donne ($x_{\min}$, $y_{\min}$) et le second indique ($x_{\max}$, $y_{\max}$). Les lignes $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$ et $y = y_{\max}$ définissent un cadre rectangulaire autour du graphe dans les figures 1 à 3. Par exemple, l'ajout de l'instruction

```
setrange(origin, whatever, whatever)
```

au fichier source de la figure 3 produit la figure 4. Le premier <coordonnées> est donné par la valeur prédéfinie et constante de type couple : `origin` et les autres coordonnées ne sont pas spécifiées. N'importe quelle variable inconnue peut aussi bien fonctionner, mais `whatever` est la représentation standard en METAPOST pour une valeur anonyme inconnue.

Il faut noter que la syntaxe de `setrange` permet de donner les valeurs des coordonnées comme des chaînes de caractères. Plusieurs commandes dans le *package* `graph` autorisent cette option. Ceci est possible parce que le langage METAPOST utilise des nombres en virgule fixe qui doivent être inférieurs à 32 768. Cette limitation n'est pas aussi sérieuse qu'il n'y paraît parce que les graphes de bonne facture imposent que les valeurs des coordonnées soient d'une « amplitude raisonnable » [2, 11]. Si l'on souhaite réellement des abscisses et des ordonnées entre 0 et 1 000 000, l'instruction

```
setrange(origin, "1e6", "1e6")
```

réalise ce travail. Toute représentation en virgule fixe ou flottante est acceptable tant que l'exposant est introduit par la lettre « `e` ».

```
draw begingraph(3in,2in) ;
 glabel.lft(btex \vbox{\hbox{Population}
 \hbox{en millions}} etex, OUT) ;
 glabel.bot(btex Âge en années etex, OUT) ;
 setrange(origin, whatever,whatever) ;
 gdraw "agepopm.d" ;
 endgraph ;
```



FIG. 4 – Graphe de la distribution d'âge de la population pour l'année 1991 et source correspondant

Le système de coordonnées n'est pas nécessairement linéaire. La commande `setcoords` permet à un ou aux deux axes d'être gradués de façon logarithmique

<système de coordonnées>→`setcoords`(<type de coordonnées>,<type de coordonnées>)
<type de coordonnées>→`log`|`linear`|`-log` |`-linear`

Un <type de coordonnées> négatif inverse l'axe x (ou y) sur la partie gauche (ou inférieure) du graphe.

La figure 5 illustre le temps d'exécution de deux algorithmes de multiplication de matrices en utilisant `setcoords(log,log)` pour spécifier une échelle logarithmique sur les deux axes. Le fichier de données matmul.d donne les temps pour les deux algorithmes :

```
20 .007861 ordinary MM : size, seconds
30 .022051
40 .050391
60 .15922
80 .4031
120 1.53
160 3.915
240 18.55
320 78.28
480 279.24

20 .006611 Strassen : size, seconds
30 .020820
40 .049219
60 .163281
80 .3975
120 1.3125
160 3.04
240 9.95
320 22.17
480 72.60
```

Une ligne blanche dans un fichier de donnée termine un ensemble de données. Des utilisations supplémentaires de la commande `gdraw` permet d'accéder à des ensembles de données supplémentaires en citant de nouveau le même fichier. Puisque chaque ligne contient une abscisse x et une ordonnée y, les éléments de commentaire après le second champ sur une ligne sont ignorés. En plaçant une commande `setcoords` entre deux instructions `gdraw` on peut tracer deux fonctions dans deux systèmes de coordonnées différents comme le montre la figure 6. Chaque fois qu'une instruction `setcoords` est utilisée, l'interpréteur METAPOST examine ce qui a été tracé, sélectionne la gamme des valeurs de coordonnées x et y et met à l'échelle tout ce qui est nécessaire. Tout ce qui est tracé après l'est fait dans un nouveau système de coordonnées qui n'a pas besoin d'avoir quoique ce soit en commun avec le précédent à moins que des commandes `setrange` imposent une gamme de valeurs similaire. Par exemple, les deux commandes `setrange` forcent les deux systèmes de coordonnées à avoir les x entre 80 et 90 et les $y \geq 0$. Lorsqu'on utilise de multiples systèmes de coordonnées, il faut spécifier où sont les étiquettes d'axe. Par défaut, les marques d'axe sont situées à gauche et sous le cadre du graphique lorsque la commande `endgraph` est interprétée. La figure 6 utilise la commande

```
autogrid(,otick.lft)
```

pour étiqueter le côté gauche du graphe avec l'ordonnée y. Cette commande agit avant la commande `setcoords`.

Ceci supprime les étiquettes d'axe par défaut, ainsi une autre commande `autogrid` est nécessaire pour étiqueter le bas et la droite du graphique utilisant un nouveau système de coordonnées.

```
draw begingraph(2.3in,2in) ;
setcoords(log,log) ;
glabel.lft(btex Secondes etex,OUT) ;
glabel.bot(btex Taille des matrices etex, OUT) ;
gdraw "matmul.d" dashed evenly ;
glabel.ulft(btex Standard etex,8) ;
gdraw "matmul.d" ;
glabel.lrt(btex Strassen etex,7) ;
endgraph ;
```

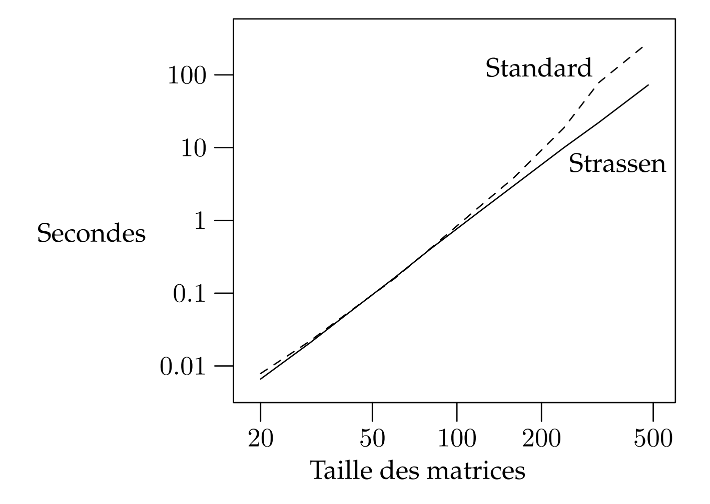


FIG. 5 – Temps d’exécution de deux algorithmes de multiplication de matrices et code METAPOST correspondant

La syntaxe générale est

`autogrid`(<commande d’étiquette d’axe>,<commande d’étiquette d’axe>)<liste d’options>

où
<commande d’étiquette d’axe>→<vide> | <marque ou grille>
<suffixe d’étiquette>
<marque ou grille>→`grid`|`itick`|`otick`

Le <suffixe d’étiquette> doit être `lft`, `rt`, `top`, `bot`. Le premier argument de `autogrid` indique comment étiqueter l’axe x et le second argument le fait pour l’axe y. Un argument <vide> supprime l’étiquetage de l’axe. Autrement, le <suffixe d’étiquette> indique de quel coté est placée l’étiquette numérique. Il faut faire attention dans l’utilisation de `top` ou `bot` pour l’axe des x et avec `lft` et `rt` pour l’axe des y. Il faut utiliser `otick` pour avoir des marques de graduation à l’extérieur, `itick` pour les mettre à l’intérieur et `grid` pour produire un quadrillage. La <liste d’options> indique comment dessiner les marques de graduation ou les lignes de quadrillage. Les lignes de quadrillage paraissent un peu forte, aussi c’est un bonne idée d’ajouter une option `withcolor` pour les griser légèrement de manière à ne pas trop charger le graphique.

```
draw begingraph(6.5cm,4.5cm) ;
   setrange(80,0, 90,whatever) ;
   glabel.bot(btex Année etex, OUT) ;
   glabel.lft(btex \vbox{ \hbox{Émissions en}
   \hbox{milliers de} \hbox{tonnes}
   \hbox{(ligneépaisse)}}etex, OUT) ;
   gdraw "lead.d" withpen pencircle scaled 1.5pt ;
   autogrid(,otick.lft) ;
   setcoords(linear,linear) ;
   setrange(80,0, 90,whatever) ;
   glabel.rt(btex \vbox{ \hbox{Microgrammes}
   \hbox{par mètre} \hbox{cube d'air} \hbox{(ligne fine)}}
   etex, OUT) ;
   gdraw "lead.d" ;
   autogrid(otick.bot,otick.rt) ;
   endgraph ;
```

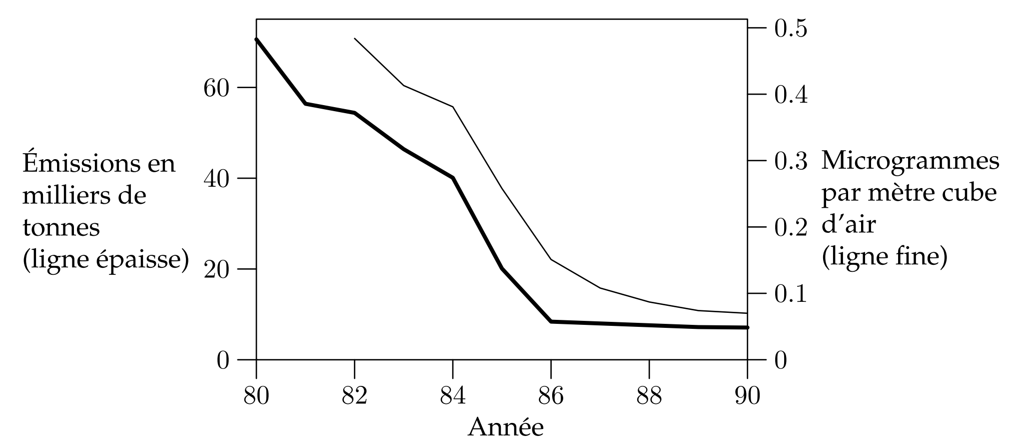


FIG. 6 – Émissions annuelles de plomb et niveau moyen enregistré aux États Unis – Le code METAPOST est indiqué au-dessus du graphe

2.3 Cadres et quadrillages en clair

Lorsque `autogrid` n'est pas suffisamment souple, les commandes d'étiquetage des axes produisent des lignes de quadrillage ou des marques de graduations, un par un. La syntaxe est

<grille ou marque>.<suffixe d'étiquette>(<format d'étiquette>,<expression chaîne ou numérique>)<liste d'option>

où <grille ou marque> et <suffixe d'étiquette> sont comme dans l'instruction `autogrid`, et le <format d'étiquette> est soit un format de chaîne comme `"%g"` soit une figure contenant l'étiquette numérique composée.

La commande d'étiquette d'axe utilise la macro

`format`(<format chaîne>,<expression chaîne ou numérique>)

pour composer des étiquettes numériques. Tous les détails apparaissent dans la section 4, mais lorsque le <format chaîne> est `"%g"`, il utilise la notation décimale à moins que le nombre soit suffisamment grand ou petit pour nécessiter la notation scientifique.

L'exemple de la figure 7 met en jeu

```
format("%g",y)
```

de façon explicite de telle sorte que les lignes de quadrillage peuvent être placées aux coordonnées transformées. Cela définit la transformation $\texttt{new}(y) = y/75 + \ln y$ et montre que cette fonction croît pratiquement linéairement[3]. Cela revient un petit peu à utiliser des coordonnées semi-logarithmiques (en y seulement), sauf que y est remplacé par $y/75 + \ln y$.

La figure 7 utilise la commande

`frame.`<suffixe d'étiquette><liste d'option>

pour tracer un cadre spécial autour du graphe. Dans ce cas le <suffixe d'étiquette> est mis à `llft` pour ne tracer que les côtés inférieur et gauche du cadre. Les suffixes `lrt`, `ulft` et `urt` tracent d'autres combinaisons de deux côtés ; les suffixes `lft`, `rt`, `top`, `bot` tracent un côté seulement et <vide> trace le cadre entier.

Par exemple

```
frame dashed evenly
```

trace les quatre côtés en pointillés. Le cadre complet est tracé par défaut lorsqu'il n'y a pas de commande `frame` explicite.

Pour étiqueter un axe, comme le fait `autogrid`, mais avec des étiquettes transformées d'une certaine manière, il faut utiliser

`auto.` ou `auto.y`

de manière à positionner les marques de graduation ou de quadrillage. Ces macros produisent des listes, avec des virgules comme séparateur, à utiliser dans les boucles `for`. N'importe quelle valeur x ou y, dans ces listes, qui ne peuvent être représentées avec précision dans le système de nombre à virgule fixe de METAPOST sont passées comme des chaînes de caractères (`string`). Un *package* de macros standard qui est chargé par l'intermédiaire de

```
input sarith
```

définit les opérateurs arithmétiques qui fonctionnent sur des nombres ou des chaînes de caractères. Les opérateurs binaires `Sadd`, `Ssub`, `Smul`, et `Sdiv` réalisent addition, soustraction, multiplication et division.

Une des applications possibles est la remise à l'échelle des données. La figure 4 utilise un fichier spécial de données `agepopm.d` dont les valeurs y sont divisées par un million. Ceci peut être évité en remplaçant « `gdraw "agepopm.d"` » par

[3]Le manuel [5] explique comment vardef définit des fonctions et comment `mlog` calcule les logarithmes.

```
vardef newy(expr y) = (256/75)*y + mlog y enddef ;
draw begingraph(3in,2in) ; glabel.lft(btex \vbox{
\hbox{Population} \hbox{en millions}} etex, OUT) ;
path p ;
gdata("ttimepop.d") ;
for y=5,10,20,50,100,150,200,250 :
   grid.lft(format("%g",y), newy(y)) withcolor .85white ;
endfor
autogrid(grid.bot,) withcolor .85white ;
frame.llft ;
endgraph ;
```

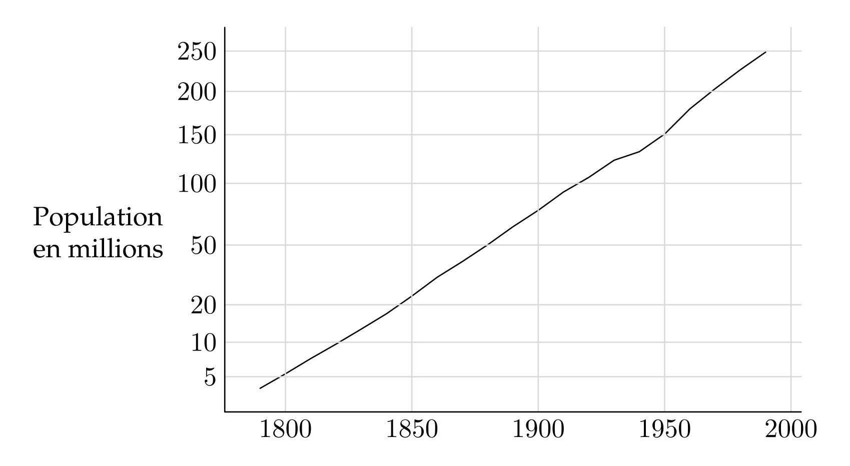


FIG. 7 – Population des États Unis en millions en fonction du temps re-exprimée suivant $p/75 + \ln p$. Le code METAPOST montré au-dessus du graphe fait référence au fichier de donnée ttimepop.d qui contient les couples de valeurs (année, $p/75 + \ln p$)

```
gdraw "agepop91.d" ;
for u=auto.y : otick.lft(format("%g", u Sdiv "1e6"), u) ;
endfor
autogrid(otick.bot,)
```

2.4 Traitement de fichiers de données

L'outil le plus général pour traiter les fichiers de données est la commande `gdata` :

`gdata`(<expression chaîne de caractères>,<variable>,<commandes>)

La commande `gdata` requiert un nom de fichier, une variable v et une liste de commandes à exécuter pour chaque ligne du fichier de données. Les commandes sont exécutées avec l'indice i fixé au numéro de ligne d'entrée et les chaînes de caractères $v1$, $v2$, $v3$... fixées pour les champs d'entrée de la ligne courante.

L'utilisation de la commande `glabel` à l'intérieur de `gdata` permet d'obtenir un nuage de points comme le montre la figure 8. Le fichier de donnée `countries.d` commence par :

```
20.910 75.7 US
1.831 66.7 Alg
```

dans lequel le dernier champ de chaque ligne indique l'étiquette à afficher.

L'utilisation de `defaultfont` dans la première ligne d'entrée sélectionne une petite fonte pour ces étiquettes. Sans ces étiquettes, aucune commande `gdata` ne serait nécessaire. Remplacer la commande `gdata` par

```
gdraw "countries.d" plot btex$\circ$etex
```

aurait changé les abréviations de pays par des cercles.

Les deux commandes `gdraw` et `gdata` ignorent un % initial et optionel de chaque ligne, analysent les champs séparés par des espaces et s'arrêtent s'ils rencontrent une ligne sans aucun champ de données. Le signe '%' est interprété par METAPOST comme un commentaire de sorte que des données numériques peuvent être placées au début d'un fichier source METAPOST. Il est souvent utile de construire une ou plusieurs courbes lorsqu'on lit un fichier de données avec `gdata`.
La commande `augment` est prévue pour ça

```
augment.<variable chemin>(<coordonnées>)
```

Si la <variable chemin> n'a pas de valeur connue, elle devient un chemin de longueur nulle pour les coordonnées choisies ; autrement un segment reliant les coordonnées est ajouté au chemin. Les <coordonnées> peuvent être une expression de type paire ou toute combinaison de chaînes de caractères et de variables de type numérique, comme l'explique le début de paragraphe 2.2.

Si un fichier `timepop.d` donne des paires t, p, `augment` peut être utilisé comme ceci pour tracer `newy`(p) en fonction de t :

```
path p;
gdata("timepop.d", s, augment.p(s1, newy(scantokens s2));); 
gdraw p;
```

La primitive de METAPOST `scantokens` interprète une chaîne de caractères comme si elle était le contenu d'un fichier source. Cette primitive retrouve la valeur numérique du champ $s2$.

La figure 9 montre comment utiliser `augment` pour lire de multiples colonnes de données et de tracer plusieurs courbes. Les chemins `p2`, `p3`, `p4`, `p5` donnent les totaux des colonnes 2 à 5 et les illustrations `lab2` à `lab5` donnent les étiquettes correspondantes. L'expression

```
picture(unfill bbox lab[j] ; draw lab[j])
```

exécute les commandes de dessin et renvoie la figure résultante. « `unfill bbox lab[j]` » permet d'obtenir un arrière-plan de couleur blanche et « `draw lab[j]` » trace l'étiquette sur le fond. La commande `gfill` est semblable à

gdraw, sauf qu'elle nécessite un chemin fermé et remplit l'intérieur avec une couleur « solide ». La couleur est noire à moins qu'une clause withcolor ne spécifie une autre couleur. Se reporter au manuel [5] pour les explications sur les boucles for, les tableaux, les couleurs et les opérateurs de construction de chemins tels que --, cycle et reverse.

```
defaultfont :="cmr7" ;
draw begingraph(3in,2in) ;
    glabel.lft(btex \vbox{ \hbox{Espérance} \hbox{de vie}}
etex, OUT) ;
    glabel.bot(btex P.N.B. par habitant (milliers de dol-
lars) etex, OUT) ;
    setcoords(log,linear)
    gdata("countries.d", s, glabel(s3, s1, s2) ;
    )
endgraph ;
```

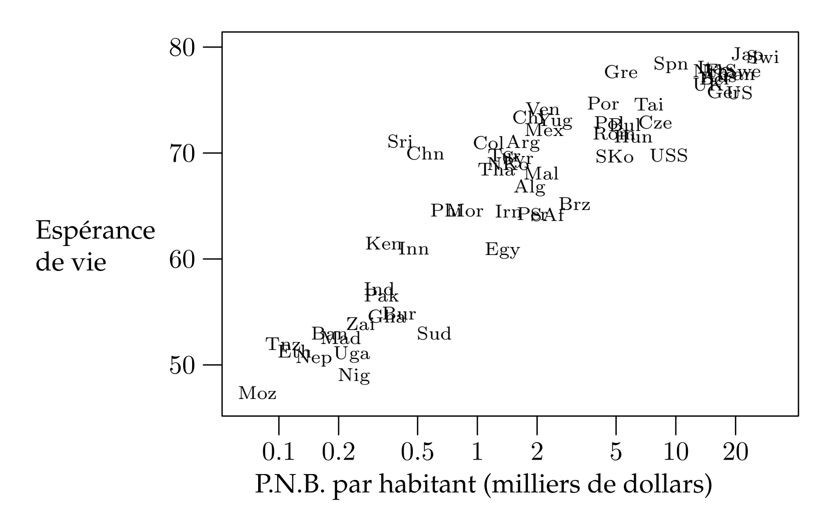


FIG. 8 – Un nuage de points et les commandes qui le créent

3 Manipulation de grands nombres

METAPOST hérite du système de numération à virgule fixe du logiciel METAFONT de KNUTH [8]. Les nombres sont exprimés comme multiples de 2^{-16} et doivent avoir une valeur absolue inférieure à 32768. KNUTH a choisi ce système parce qu'il est parfaitement adapté à la définition des fontes et garantissait des résultats identiques sur tous les types d'ordinateur. Les nombres à virgule fixe posent rarement des problèmes dans METAPOST parce que les calculs sont basés sur des coordonnées qui sont limitées par la taille du papier sur lequel les sorties doivent être imprimées. Ceci ne concerne pas les données d'entrée pour les tracés de graphes. Bien que les graphes soient plus présentables lorsque les axes sont gradués avec des valeurs raisonnables (pas trop grandes), la stricte limite de l'arithmétique serait un inconvénient.

Un moyen simple pour manipuler des grands nombres est d'inclure la ligne

```
input sarith
```

et d'utiliser les opérateurs binaires `Sadd`, `Ssub`, `Smul` et `Sdiv` à la place de `+`, `-`, `*` et `/`. Ces opérateurs sont inefficaces mais d'une grande souplesse. Ils acceptent des nombres ou des chaînes de caractères et renvoient des chaînes de caractères en notation exponentielle avec l'exposant marqué par « `e` » ; par exemple, `"6.7e-11"` signifie 6.7×10^{-11}.

```
draw begingraph(3in,2in) ;
   draw begingraph(3in,2in) ;
    glabel.lft(btex \vbox{\hbox{Quadrillions}
    \hbox{de BTU}} etex, OUT) ;
    path p[] ;
    numeric t ;
    gdata("energy.d",$,
      t :=0 ; augment.p1($1,0) ;
      for j=2 upto 5 :
        t :=t+scantokens $[j] ; augment.p[j]($1,t) ;
      endfor)
    picture lab[] ;
    lab2=btex charbon etex ;
    lab3=btex fuel etex ;
    lab4=btex gaz naturel etex ;
    lab5=btex hydoélectricité etex ;
    for j=5 downto 2 :
      gfill p[j]-reverse p[j-1]-cycle withcolor
.16j*white ;
    glabel.lft(picture(unfill bbox lab[j] ;
    draw lab[j]), .7+length p[j]) ;
    endfor
    endgraph ;
```

FIG. 9 – Un graphe de la production annuelle d'énergie des États Unis et les commandes qui le créent

L'opérateur unaire[4]

`Sabs`<string>

renvoie une chaîne de caractères représentant la valeur absolue. Les opérateurs `Sleq` et `Sneq` réalisent des comparaisons numériques sur des chaînes de caractères et renvoient un résultat booléen.

L'opérateur

`Scvnum`<string>

renvoie la valeur numérique pour une chaîne de caractère si cela peut être fait sans provoquer un dépassement de capacité du système numérique à virgule fixe de METAPOST. Si la chaîne de caractères ne contient pas le caractère « e », il est plus efficace d'utiliser la primitive

`scantokens`<string>

Les opérateurs précédents sont basés sur un module de bas niveau qui manipule les nombres en « format `Mlog` ». Un nombre x en format `Mlog` représente

$$\mu^{2^{16}x}, \text{où } \mu = -e^{2^{-24}}$$

Toute valeur entre $1,61 \times 10^{-28}$ et $3,88 \times 10^{55}$ peut être représentée de cette manière. (Il existe une constante `Mten` telle que k`*Mten` représente 10^k pour tout entier k dans l'intervalle [-29, 55]).

La principale raison de mentionner le format `Mlog` est qu'il permet de manipuler les données d'un graphe comme un chemin de METAPOST. La fonction

`Mreadpath`(nom de fichier)

lit un fichier de données et renvoie un chemin dans lequel toutes les coordonnées sont dans le format `Mlog`. Une variable interne `Gpaths` détermine si `gdraw` et `gfill` attendent des chemins dans le format `Mlog`. Par exemple, les instructions suivantes tracent le graphe avec les données du fichier `agepop91.d`, les y étant divisées par un million :

```
interim Gpaths :=log ;
 gdraw Mreadpath("agepop91.d") shifted (0,-6*Mten) ;
```

4 Composition des nombres

Le *package* `graph` a besoin de calculer les étiquettes d'axes et ensuite de les composer. La macro

`format`(<expression chaîne de caractères>,<numérique ou expression chaîne de caractères>)

[4]L'argument d'un opérateur unaire ne nécessite pas d'être entre parenthèses à moins qu'il ne s'agisse d'une expression mettant en jeu un opérateur binaire.

réalise cette tâche. Il faut d'abord utiliser l'instruction `input graph` ou `input format` pour charger le fichier de macro. La macro requiert un argument de format chaîne de caractères et un nombre et renvoie une figure contenant le résultat composé. Ainsi

`format("%g",2+2)` produit 4

et

`format("%3g","6.022e23")` produit $6,02 \times 10^{23}$

Un format chaîne de caractères consiste en :
– un caractère initial optionnel ne contenant pas le signe % ;
– un signe % ;
– un paramètre optionnel de précision, noté p ;
– une des lettres de conversion `e`, `f`, `g`, `G` ;
– un caractère final optionnel β.

Les caractères initial et final sont composés dans la fonte par défaut (généralement `cmr10`) et le nombre composé est placé entre eux deux. Pour les formats `e` et `g`, la précision p est le nombre de chiffres significatifs autorisés après arrondi, pour les formats `f` et `G`, le nombre est arrondi au multiple de 10^{-p} le plus proche. Si la précision n'est pas indiquée, $p = 3$ est pris par défaut. Le format `e` utilise toujours la notation scientifique et le format `f` utilise la notation décimale ordinaire si le nombre est strictement inférieur à 1 000, sinon c'est le format scientifique qui est automatiquement utilisé. Les formats `g` et `G` utilisent automatiquement la notation scientifique pour les nombres non nuls inférieurs à 0,001.

La macro `format` nécessite un ensemble de modèles pour déterminer quelle fonte utiliser, comment positionner l'exposant... Les modèles sont normalement initialisés automatiquement, mais il est possible de les fixer explicitement en passant cinq arguments de type figure (*picture*) à la macro `init_numbers`. Par exemple, la définition par défaut pour les utilisateurs de TEX est

```
init_numbers(btex$-$etex, btex$1$etex, btex${\times}
      10$etex, btex${}^-$etex, btex${}^2$etex)
```

Le premier argument dit comment composer un signe $-$; le second est un exemple de mantisse d'un chiffre ; le troisième indique le symbole à mettre après la mantisse en notation scientifique ; les suivants indiquent le signe $-$ de l'exposant et l'exemple d'un exposant à un chiffre.

La variable figure `Fe_plus` permet d'indiquer le signe $+$ pour des nombres positifs et `Fe_base` indique le symbole qui précède l'exposant quand on compose une puissance de 10. L'appel de `init_numbers` initialise `Fe_plus` à une figure vide et construit `Fe_plus` à partir de ses deuxième et troisième arguments.

5 Conclusion

Le *package* `graph` facilite la génération de graphes en langage METAPOST. Les bénéfices premiers sont la puissance de METAPOST et sa capacité à interagir avec TEX ou troff pour composer les étiquettes. Les étiquettes ainsi

composées peuvent être stockées dans des variables figure et manipulées de différentes manières. On peut ainsi mesurer la *bounding box* et fixer un fond blanc par exemple.

On a vu comment créer des zones ombrées et contrôler l'épaisseur de lignes, leur couleur ainsi que le style des lignes pointillées. De nombreuses autres variations sont possibles. Le langage complet METAPOST[5] procure beaucoup d'autres caractéristiques potentiellement utiles. Il possède aussi suffisamment de puissance de calcul pour être utile à la génération et au traitement des données.

A Résumé du *package* `graph`

Dans les descriptions qui suivent, les lettres italiques telles que w et h dénotent des expressions de type paramètre et les mots entre crochets dénotent d'autres éléments syntaxiques.

Sans autre spécification, les expressions de type paramètre peuvent être soit des nombres ou des chaînes de caractères. Une <liste d'options> est une liste d'options de dessin telles que `withcolor, .5white` ou `dashed evenly`; un <suffixe d'étiquette> est l'un des éléments parmi `lft`, `rt`, `top`, `ult`, `urt`, `llft`, `lrt`.

A.1 Gestion des graphes

`begingraph`(w,h) commence un nouveau graphe dont la largeur et la hauteur sont données par les paramètres w et h.

`endgraph` termine un graphe et renvoie la figure courante.

`setcoords`(t_x,t_y) met en place un nouveau système de coordonnées spécifié par les indicateurs t_x et t_y. Les valeurs de ces indicateurs sont ±`linear` (linéaire) et ±`log` (logarithmique).

`setrange`(<coordonnées>,<coordonnées>) fixe les limites inférieures et supérieures pour le système de coordonnées courant. Chaque <coordonnées> peut être une simple expression de type paire ou deux valeurs numériques ou une expression de type chaîne de caractères.

A.2 Tracé et étiquettes

Toutes les commandes de tracé et d'étiquetage peuvent être suivies par une <liste d'options>. À côté des options usuelles de tracé de METAPOST, la liste peut contenir une clause `plot` (de type figure) pour tracer un dessin à chaque point.

Les commandes de tracé et d'étiquetage sont étroitement reliées à un ensemble de commandes de `plain` METAPOST dont le nom est similaire. Les commandes `gdrawarrow` et `gdrawdblarow` sont incluses pour maintenir cette similarité.

`gdotlabel`.<suffixe d'étiquette>(p,<localisation>) est identique à `glabel` sauf que cette commande ajoute aussi un point à l'endroit de l'étiquette.

`gdraw` p trace le chemin p ou, si p est une chaîne de caractères, lit les couples de coordonnées dans le fichier p et trace une ligne polygonale par ces points.

`gdrawarrow` p réagit de la même façon que `gdraw` sauf que la commande ajoute une tête de flèche à la fin du chemin.

`gdrawdblarrow` p réagit de la même façon que `gdraw` sauf que la commande ajoute une tête de flèche au début et à la fin du chemin.

`gfill` p remplit un chemin fermé p ou lit les coordonnées dans le fichier nommé par la chaîne de caractères p et remplit la ligne polygonale résultante.

`glabel`.<suffixe d'étiquette>(p,<localisation>) Si p n'est pas un dessin, p doit être une chaîne de caractères. Cette commande compose p en utilisant `defaultfont`, puis place p à proximité du lieu d'insertion et le décale de la quantité spécifiée par <suffixe d'étiquette>. Le lieu d'insertion, <localisation>, peut être des coordonnées x et y, un couple donnant x et y, une valeur numérique indiquant le paramètre t (*time*) sur le dernier chemin tracé, ou `OUT` pour poser une étiquette à l'extérieur du graphe.

A.3 Quadrillage, graduations et cadres

`auto`.< x ou y > produit les graduations x ou y par défaut.

`autogrid` (<commande d'étiquetage d'axe>, <commande d'étiquetage d'axe>) trace les étiquettes par défaut des axes en utilisant les commandes spécifiées pour les axes x et y. Une <commande d'étiquetage d'axe> peut être <vide> ou contenir `itick`, `otick` ou `grid` suivi par un <suffixe d'étiquette>.

`frame`.<suffixe d'étiquette> <liste d'options> trace un cadre autour du graphe ou trace la partie du cadre spécifiée par le <suffixe d'étiquette>.

`grid`.<suffixe d'étiquette>(f, z) trace un quadrillage à travers le graphe à partir du côté spécifié par le <suffixe d'étiquette> et l'étiquette en utilisant le format caractère f à l'endroit indiqué par la valeur de coordonnées z. Si f est une figure, la commande indique l'étiquette.

`itick`.<suffixe d'étiquette>(f, z) équivaut à la commande `grid` sauf que cette commande trace une graduation dirigée vers l'intérieur du graphe.

`otick`.<suffixe d'étiquette>(f, z) équivaut à la commande `grid` sauf que cette commande trace une graduation dirigée vers l'extérieur du graphe.

A.4 Commandes diverses

`augment`.<variable>(<coordonnées>) ajoute les <coordonnées> au chemin stocké dans la <variable>.

`format`(f, x) compose x suivant le format f et renvoie la figure qui en résulte.
`gdata`(f, <variable>, <commandes>) lit le fichier nommé par la chaîne de caractères f et exécute les <commandes> pour chaque ligne en utilisant la <variable> comme un tableau pour stocker les données.

`init_numbers`(s, m, x, t, e) procure cinq figures pour les futures opérations `format` : s correspond au signe $-$; m est un exemple de mantisse ; x suit la mantisse ; t est le signe $-$ de l'exposant e.

`Mreadpath`(f) lit un chemin pour le fichier de données nommé par le format f et le renvoie en « format `Mlog` ».

A.5 Arithmétique sur les chaînes numériques

Il est nécessaire d'utiliser l'instruction `input sarith` avant d'utiliser les macros suivantes :

`Sabs` x calcule $|x|$ et renvoie une chaîne numérique.

x `Sadd` y calcule $x + y$ et renvoie une chaîne numérique.

`Scvnum` x renvoie la valeur numérique pour la chaîne de caractère x.

x `Sdiv` y calcule x/y et renvoie une chaîne numérique.

x `Sleq` y renvoie le résultat booléen de la comparaison $x \leq y$.

x `Smul` y calcule $x * y$ et renvoie une chaîne numérique.

x `Sneq` y renvoie le résultat booléen de la comparaison $x \neq y$.

x `Ssub` y calcule $x - y$ et renvoie une chaîne numérique.

A.6 Variables internes et constantes

`Autoformat` format des chaînes de caractères utilisé par `autogrid`. Par défaut : `"%g"`.

`Fe_base` ce qui précède l'exposant lorsqu'on compose une puissance de dix.

`Fe_plus` représente le signe + initial pour les exposants positifs.

`Gmarks` nombre minimum de marques de graduations par pour `auto` et `autogrid`. Par défaut : 4.

`Gminilog` rapport minimum entre l'espace le plus grand et le plus petit pour une échelle logarithmique pour `auto` et `autogrid`. Par défaut : 3.0.

`Gpaths` code pour les coordonnées utilisé dans les chemins `gdraw` et `gfill` : `linear` dans la forme standard, `log` pour le « format `Mlog` ».

`Mten` le « format `Mlog` » pour 10.0.

B Nouvelles caractéristiques du langage

Les macros `graph.mp` et les routines arithmétiques dans les fichiers `marith.mp` et `sarith.mp` utilisent différentes caractéristiques du langage qui ont été introduites dans la version 0.60 de METAPOST. Elles sont résumées ici parce qu'elles ne sont pas intégrées dans la documentation existante [5, 6]. Est également nouvelle la macro

`picture`(<commandes de tracé>)

qui a été utilisée dans le paragraphe 2.4 pour trouver la figure produite par une séquence de commandes de tracé.

B.1 Lecture et écriture des fichiers

Un nouvel opérateur

`readfrom` <nom de fichier>

renvoie une chaîne de caractères donnant la ligne suivante du fichier cité. Le <nom de fichier> peut être n'importe quelle expression primaire de type chaîne de caractères. Si le fichier est terminé ou ne peut pas être lu, le résultat est une chaîne de caractères consistant en un simple caractère nul (ou vide). Le *package* de macro `plain` préchargé introduit le nom `EOF` pour cette chaîne de caractères. Après que `readfrom` a renvoyé `EOF`, de nouvelles lectures à partir du même fichier provoque une relecture à partir du début du fichier.

La commande opposée à `readfrom` est la commande

`write`<expression chaîne de caractères> `to` <nom de fichier>

Cette commande écrit une ligne de texte dans le fichier de sortie spécifié, en ouvrant d'abord le fichier, si nécessaire. Tous ces fichiers sont fermés automatiquement quand le programme se termine. Ils peuvent être fermés explicitement en utilisant `EOF` comme <expression chaîne de caractères>. Le seul moyen pour savoir si la commande `write` a été réalisée avec succès est de fermer le fichier et d'employer readfrom pour lire son contenu.

B.2 Extraction d'information dans les dessins

Les images *(pictures)* METAPOST sont composées de lignes, de contours colorés, de morceaux de textes composés, de chemins de délimitation (ou de détourage) et de chemins `setbounds`. Un chemin `setbounds` donne une *bounding box* artificielle comme le nécessite la sortie TEX. Une figure peut posséder plusieurs composants de chaque type. Il est possible d'y accéder au moyen d'une itération de la forme

`for` <token symbolique> `within` <expression figure> : <texte de boucle> `endfor`

Le <texte de boucle> peut être n'importe quel texte équilibré entre `for` et `endfor`. Le <token symbolique> est une variable de boucle qui scrute les

composants de la figure dans l'ordre où ils ont été tracés. Le composant pour un chemin de délimitation ou `setbounds` comprend tout ce que le chemin inclut. Ainsi, si une simple délimitation ou un chemin `setbounds` s'applique à tout ce qui est dans la figure, la figure entière peut être considérée comme un seul et gros composant. De manière à recenser les composants d'une telle figure, l'itération `for...within` ignore les chemin de délimitation ou `setbounds` dans ce cas.

Une fois que l'itération `for...within` a trouvé un élément figure, il existe de nombreux opérateurs pour l'identifier et extraire l'information pertinente. L'opérateur

`stroked`<expression primaire>

teste si l'expression est une figure connue dont le premier composant est une ligne. De façon similaire, les opérateurs `filled` et `textual` retournent la valeur booléenne vraie *(true)* si le premier composant est un contour coloré ou un morceau de texte composé. Les opérateurs `clipped` et `bounded` testent si l'argument est une figure connue qui commence par un chemin de détourage ou de `setbounds`. Ceci est vrai si le premier élément est détouré ou entouré ou si la figure est entourée par un chemin de détourage ou de `setbounds`.

Il existe également de nombreux opérateurs d'extraction qui testent le premier élément d'une figure. Si `p` est une figure et si `stroked p` est vrai, `pathpart p` est le chemin décrivant la ligne qui est esquissée, `penpart p` est le stylo qui a été utilisé, `dashpart p` est le motif de pointillé et la couleur est

(`redpart p`, `greenpart p`, `bluepart p`)

Si la ligne n'est pas pointillée, `dashpart p` renvoie une figure vide.

Les mêmes opérateurs d'extraction fonctionnent lorsque `filled p` est vrai, excepté que `dashpart p` ne présente aucun intérêt dans ce cas. Pour les éléments textes, `textual p` est vrai, `textpart p` renvoie le texte qui a été composé, `fontpart p` indique la fonte qui a été utilisée et `xpart p`, `ypart p`, `xxpart p`, `xypart p`, `yxpart p`, `yypart p` indiquent de combien le texte a été décalé ou tourné ainsi que le facteur d'échelle. Les opérateurs (`redpart p`, `greenpart p` et `bluepart p`) fonctionnent également sur les éléments textes.

Lorsque `clipped p` ou `bounded p` est vrai, `pathpart` indique le chemin de détourage ou le chemin `setbounds` et les autres opérateurs d'extraction n'ont pas d'intérêt. De telles extractions sans intérêts ne produisent pas d'erreurs, elles retournent seulement des valeurs nulles : le chemin trivial `(0,0)` pour `pathpart`, `nullpen` pour `penpart`, une figure vide pour `dashpart`, zéro pour `redpart`, `greenpart`, `bluepart` et la chaîne de caractères vide pour `textpart` ou `fontpart`.

Un dernier opérateur pour extraire des informations d'une figure est

`length`<figure primaire>

Cette commande retourne le nombre d'éléments qu'une itération `for...within` devrait trouver.

B.3 Autres nouvelles caractéristiques

Les *packages* `marith.mp` et `sarith.mp` utilisent des nombres supérieurs à 4 096. Puisque de telles valeurs peuvent causer des problèmes d'*overflow* (de dépassement de capacité) dans le processus de résolution des équations linéaires dans METAPOST et les algorithmes d'ajustement de courbes, ces valeurs ne sont normalement autorisées que pour les résultats intermédiaires. Cette limitation est supprimée lorsque la variable `warningcheck` vaut zéro. Dans les versions antérieures de METAPOST, cette limitation pouvait être supprimée uniquement pour les variables. Les constantes explicites devaient toujours être strictement inférieures à 4 096. Pour être complet, il faut aussi mentionner une autre des nouvelles caractéristiques de la version 0.60 de METAPOST. Lorsque le matériel TEX est inclus dans une figure par l'instruction `btex...etex`, l'épaisseur des filets horizontaux et verticaux est arrondie exactement au nombre de pixels ; c'est-à-dire les sorties interprétées par METAPOST selon les règles de conversion POSTSCRIPT [7] font que la largeur de pixel est égale à la valeur supérieure de la largeur non arrondie. En fait, une relation similaire est encore valable pour toutes les épaisseurs de ligne. POSTSCRIPT fixe les largeurs de ligne d'abord en les transformant en coordonnées du périphérique puis en les arrondissant de façon appropriée.

Références

[1] Jon L. BENTLEY et Brian W. KERNIGHAN, Grap – a language for typesetting graphs In *Unix Research System Papers*, volume II ; pages 109-146. AT&T Bell Laboratories, Murray Hill, New Jersey, 20th edition, 1990.

[2] William S. CLEVELAND, *The Elements of Graphing Data* Hobart Press, Summit, New Jersey, 1985.

[3] William S. CLEVELAND, A model for studying display methods of statistical graphics (with discussion)
Journal of Computational and Statistical Graphics, 3, à paraître.

[4] William S. CLEVELAND, *Visualizing Data* Hobart Press, Summit, New Jersey, à paraître.

[5] J. D. HOBBY, A user's manual for METAPOST *Computing Science Technical Report n° 162*, AT&T Bell Laboratories, Murray Hill, New Jersey, April 1992. Peut être obtenu par courrier "`send 162 from research/cstr`" à "`netlib@research.att.com`"
Version française : Un manuel d'utilisation pour METAPOST (ce cahier)

[6] John D. HOBBY, Introduction to METAPOST In *EuroTEX'92 Proceeding*, pages 21-36, september 1992.

[7] Adobe Systems Inc., *PostScript Language Reference Manual* Addison Wesley, Reading, Massachusets, 2nd edition, 1990.

[8] D. E. KNUTH, *METAFONT the Program* Addison Wesley, Reading, Massachusets, 1986. Volume D of *Computers and typesetting*.

[9] Leslie LAMPORT, *LATEX : A Document Preparation System* Addison Wesley, Reading, Massachusets, 1986.

[10] U.S. Bureau of the Census, *Statistical Abstracts of the United States :* 1992 Washington, D.C., 112th edition, 1992.

[11] Edward R. TUFTE, *Visual Display of Quantitative Information* Graphics Press, Box 430, Cheshire, Connecticut 06410, 1983.